######################################################################

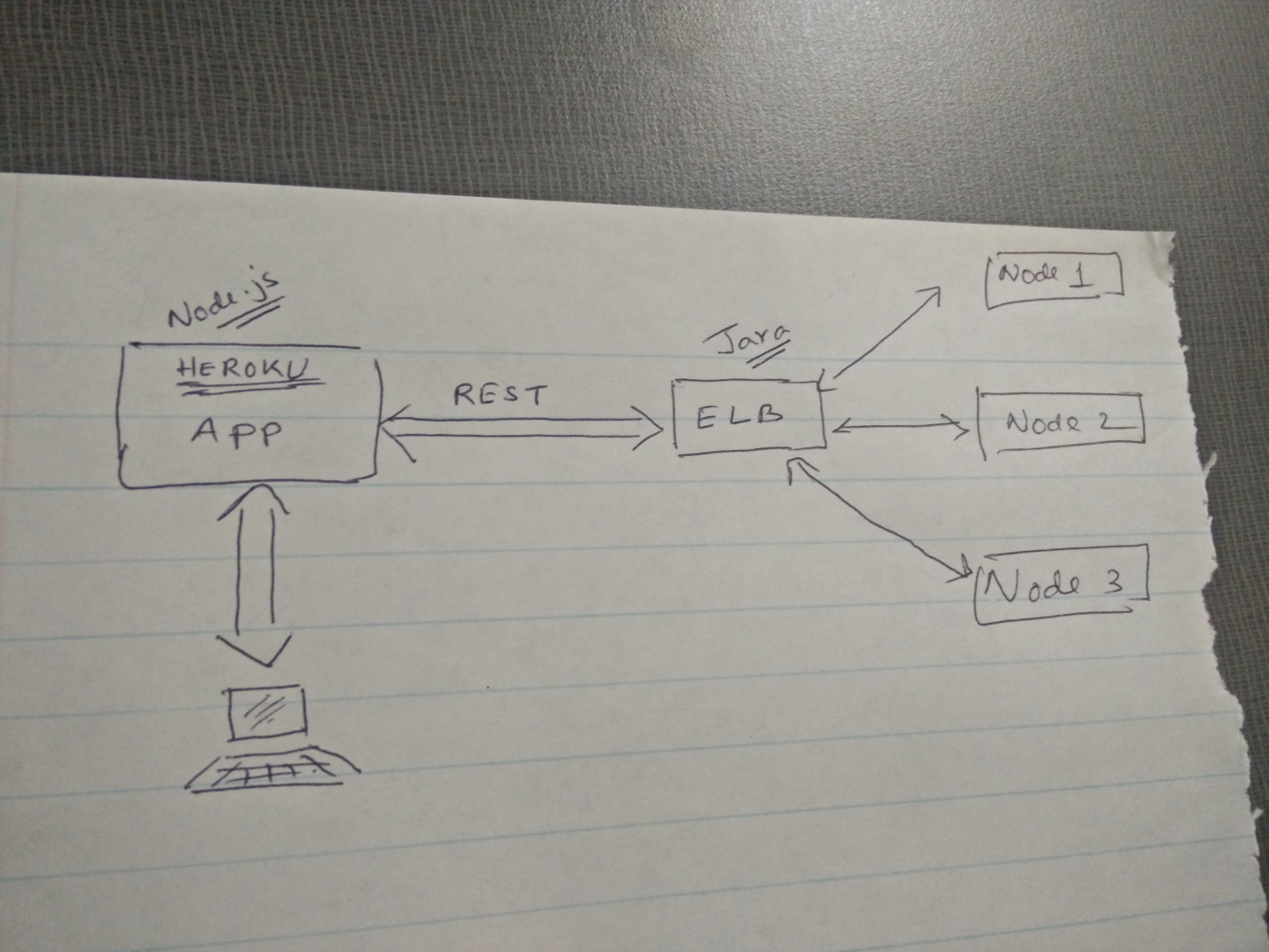
# AUTHOR: Rakesh Datta

# DATE : April 9, 2016

**GIT Repository**: <https://github.com/sjsu-cmpe281/shoppingcart>

**Waffle Link** : <https://waffle.io/sjsu-cmpe281/shoppingcart>

TASK 1: Conceived the high-level Product Architecture and Flow.



As a result of the design thinking phase, I have understood the high-level design.

Heroku is a PaaS where we design, deploy and host our front-end. HTML5 is used to design the Views and JS is used to design the controller.

These things run on top of node.js.

1. When a HTTP request (CRUD) comes to the App from external user, if this request is regarding static pages then the pages are sent back to user directly. Otherwise, it is handled by the relevant App Controller Function, which then sends a new HTTP request to the AWS ELB. In other terms, the App Controller invokes a REST API.
2. AWS ELB forwards the request to one of the AWS EC2 Nodes. Depending on the Requested method (POST/GET/PUT/DELETE), the node then invokes a handler. The handler will execute some mongodb/riak operation and send the response back over HTTP.
3. Node.js in Heroku receives the response back, manipulate it and updates the View with the results.

We are going to design a Product Catalog with a list of items. This will be achieved through mongoDb. When a user selects certain item(s), it is added to the shopping cart which is implemented in Riak. This shopping cart may be a perpetual one or a persistent one (tied to the username; user should be able to see the shopping cart across multiple sessions). At minimum the shopping cart data should be persistent across a single session.

We are yet to discuss and close on this.

TASK 2: Implemented a Java Based HTTP server that exposes REST API.

To start with, I have designed a Java Based HTTP server and exposed REST APIs as part of it. I used *HttpServer* library to create the server. It is able to receive external HTTP requests over port 80. The nodes which run this server would talk to each other over 8080 for replication and administration purpose.

The above mentioned Java Code imports mongo libraries. It would also instantiate a Mongo Client and Mongo Server on startup. Post that mongo configuration should take place which will take care of creating database/table etc(this part is in progress and partially completed).

The REST handlers should use the mongo-client API to execute operations on mongoDb.

<https://github.com/sjsu-cmpe281/shoppingcart/blob/master/server/src/SMHttpServer.java>

TASK 3: Implemented a simple Java based Mongo Client.

To understand the mongo operations a simple Java based mongo client is implemented. This code spawns a mongo client, sets up a connection with the mongo server and sends a simple request from the client to the server.

This is under test. Once this code is full-proof and working it will be used for the REST APIs.

e.g. <https://github.com/sjsu-cmpe281/shoppingcart/blob/master/server/src/MyMongoClient.java>

*MongoClient mongoClient = new MongoClient( "localhost" , 27017 );*

*DB db = mongoClient.getDB( "cmpe281" );*

*boolean auth = db.authenticate(UserName, Password);*

TASK 4: Deployed a sample Heroku App.

In order to understand the Heroku infrastructure and how will it communicate with the back-end, a sample app is deployed. Currently investigation is going on to understand how the controllers will send HTTP requests to the back-end. After that we will deploy a simple page a simple controller and push a request to the back-end to setup the connection between Heroku and AWS EC2 Instances.

TASK 5: Researched MongoDB use case for product catalog.

To implement a product catalog, there are several drawbacks of RDMS.

Therefore, to make the database simpler and reduce the cost and response time of operations, mongoDb is used for designing modern-day e-commerce product catalog. MongoDB helps in avoiding unnecessary Joins, multiple table inheritance etc.

TASK 6: Infrastructure Work: GitHub & Waffle.

To collaborate and work on the same and updated code, I have setup a GIT repository. Have brainstormed and decided the directory structure.

The whole project directory follows a heroku App directory structure. On top of that two more folders are added.

‘server’ : To add back-end code

‘misc’ : To add non-functional items like docs, images etc.

Team just need to ‘git clone’ this project and start working on code. Back-end team will be worked on the code residing in the ‘server’ folder. Front-end team will work on rest of the directories (html and controllers).

TASK 7: Team Consultation.

Helping the whole team with technical consultations. Breaking up the tasks and deciding the owners. Helping the owners to conceive the product, mock-ups, design the road-map, remove blockers etc.